Informe laboratorio numero 3

Problema:
Consolidar en una aplicación los datos de algunos mercados de divisas y de acciones internacionales.

Requerimientos Funcionales:
R1: Eliminar o modificar Datos
R2: Consultar el precio más alto de una acción o un mercado de divisas en un rango de tiempo dada una fecha inicial y una fecha final.
R3: Consultar el precio más bajo de una acción o un mercado de divisas en un rango de tiempo dada una fecha inicial y una fecha final.
R4: Consultar el periodo de tiempo donde una acción / mercado de divisas tuvo su mayor crecimiento**.**
R5: Mostrar una gráfica del estado de los precios de una acción / mercado de divisas. En la gráfica debe ser posible agregar hasta un máximo de 3 acciones / mercados de divisas en donde cada indicador de gráfica deberá tener un color diferente.
R6: Mostrar las acciones / Mercado de divisas superan un valor en un rango de tiempo.
R7:Mostrar las 3 acciones / Mercados que presentaron mayor crecimiento en un rango de tiempo.

Requerimientos no funcionales:
Usabilidad:
El usuario solo contará en la interfaz los espacios correspondientes para cargar los datos que tiene que tener la información correspondiente al programa para que funcione correctamente de lo contrario cualquier tipo de error al cargar los datos no se podrá hacer las de búsqueda ,eliminación y demás requerimientos requeridos en la aplicación.

Recopilacion de informacion:
Para resolver este problema y ayudar a la BVC a consolidar los datos de acciones y acciones de divisas internacionales se tuvo que investigar tales significados y cómo funcionaban tales procesos para acercarnos al problema y poder darle solución de una manera eficaz.

Oferta:

La oferta es la cantidad de productos o servicios ofrecidos en el mercado. En la oferta, ante un aumento del precio, aumenta la cantidad ofrecida.

Demanda:
La demanda es la cantidad de bienes o servicios que los compradores intentan adquirir en el mercado.
https://www.elblogsalmon.com/conceptos-de-economia/que-es-la-oferta-y-la-demanda

Mercados financieros

Los mercados financieros es un espacio que puede ser físico o virtual, a través del cual se intercambian activos financieros entre agentes económicos y en el que se definen los precios de dichos activos.

Los mercados financieros conforman un espacio cuyo objetivo es canalizar el ahorro de las familias y empresas a la inversión. De tal manera que las personas que ahorran tengan una buena remuneración por prestar ese dinero y las empresas puedan disponer de ese dinero para realizar inversiones.
https://economipedia.com/definiciones/mercados-financieros.html

Mercado de Divisas:

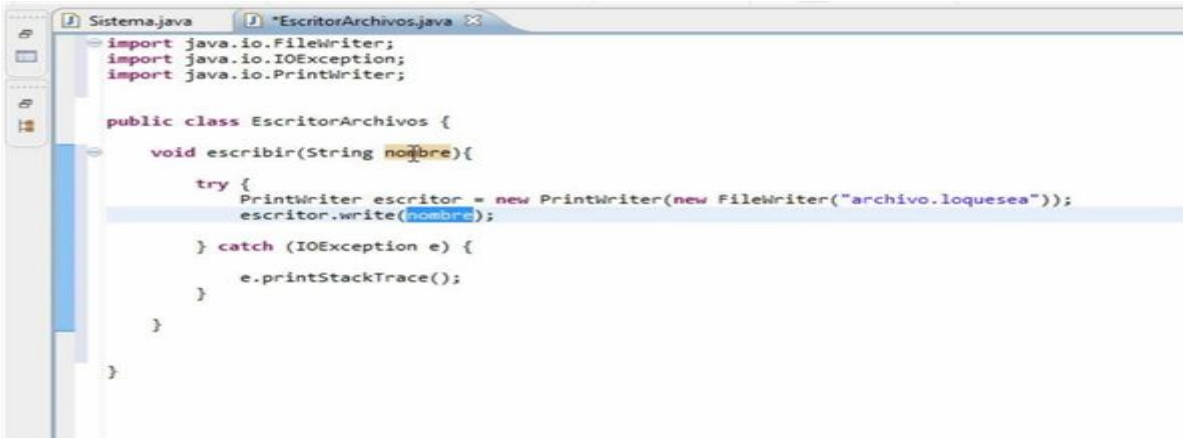
El mercado de divisas o mercado de tipos de cambio es un mercado global y descentralizado en el que se negocian divisas y que nació con el objetivo de facilitar cobertura al flujo monetario que se deriva del comercio internacional.
https://www.bbva.com/es/mercado-divisas-que-es-como-funciona/

Estructuras de datos eficientes:

En las estructuras de datos eficientes se trabaja el análisis algorítmico (costes, casos, notación asintótica, resolución de recurrencias simples), la especificación y la verificación formal de algoritmos recursivos simples. Se introduce el análisis de la eficiencia de los programas como un criterio más de calidad. Se estudia también el diseño de algoritmos recursivos simples y múltiples, y varias técnicas de resolución de problemas combinatorios. - En la segunda parte, se introduce el concepto y especificación de tipos abstractos de datos como estructuras de datos lineales, árboles, tablas y grafos. Las técnicas básicas mencionadas incluyen conocimientos teóricos y prácticos, habilidades, experiencias y sentido crítico, todas ellas fundamentadas en teorías y técnicas sólidas, comprobadas y bien establecidas.

Lectura y escritura de archivos:

Podemos utilizar diferentes formas de guardar los datos dentro de un programa, la más común es la lectura y escritura de archivos, estos pueden ser de texto(.txt) o archivos un poco más complejos como hojas de excel, archivos en bases de datos online, archivos .CSV, entre muchos otros. Se utilizan algoritmos especiales para leer estos archivos, y otros algoritmos para poder escribir sobre estos, y que la información sea persistente. Ejemplo: Para escribir archivos podemos usar este tipo de algoritmo:



Y para leer los archivos podemos utilizar el siguiente:

Fase 3. Búsqueda de soluciones creativas

Árbol Binario de Búsqueda(ABB):

La búsqueda en árboles binarios es un método de búsqueda simple, dinámico y eficiente considerado como uno de los fundamentales en Ciencia de la Computación. De toda la terminología sobre árboles,tan sólo recordar que la propiedad que define un árbol binario es que cada nodo tiene a lo más un hijo a la izquierda y uno a la derecha.Para construir los algoritmos consideraremos que cada nodo contiene un registro con un valor clave a través del cual efectuaremos las búsquedas.En las implementaciones que presentaremos sólo se considerará en cada nodo del árbol un valor del tipo t Elemento aunque en un caso general ese tipo estará compuesto por dos:una clave indicando el campo por el cual se realiza la

ordenación y una información asociada a dicha clave o visto de otra forma,una información que puede ser compuesta en la cual existe definido un orden.

Es un árbol binario en el cual se cumple que para cada nodo x:

- los nodos del subárbol izquierdo son menores o iguales a x.
- los nodos del subárbol derecho son mayores o iguales a x.

Propiedades:

Sea x un nodo del árbol:

- Si y es un nodo en el subárbol izquierdo de x , entonces key[y] ≤ key[x].
- Si y es un nodo en el subárbol derecho de x, entonces key[y] ≥ key[x].

Importancia ABB:

- Operaciones básicas como insertar, borrar y buscar, toman un tiempo proporcional a la altura del árbol.
- Para un árbol binario perfecto con n nodos, las operaciones básicas toman Θ(log n)
- Si el árbol se construye como una cadena lineal de n nodos, tomarían Θ(n)

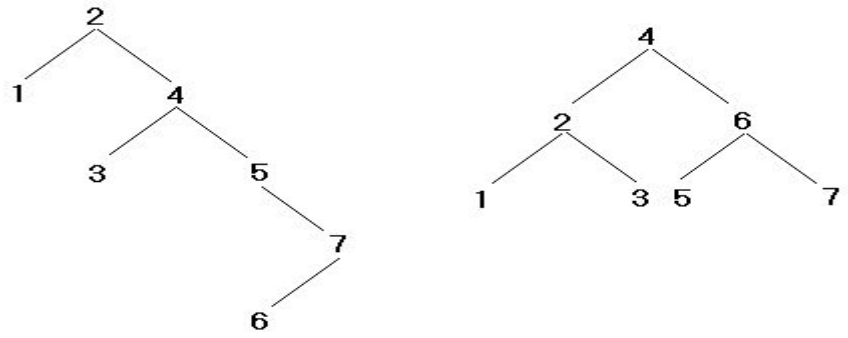


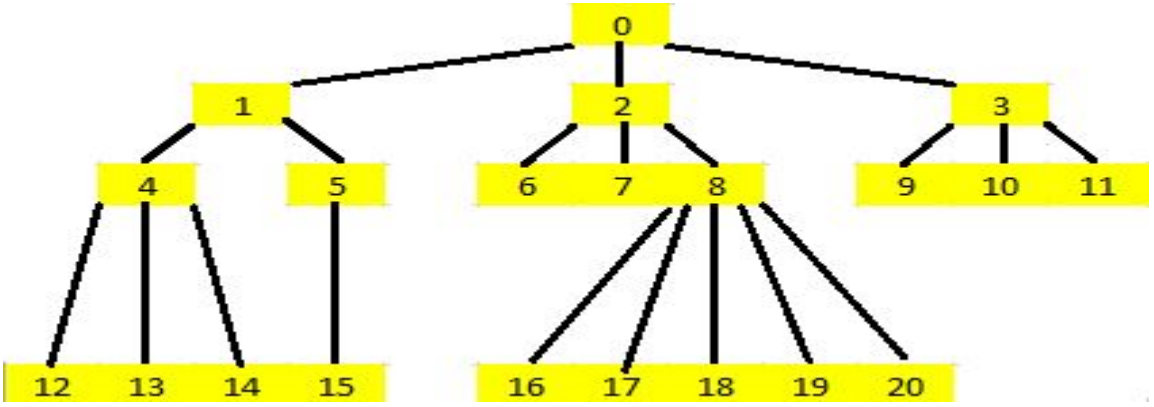
Figura 1:Dos ABB con los mismos elementos.

http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_BB.htm

Árbol N-ARIO.

Que es?

- Es una estructura recursiva en la que cada elemento puede tener cualquier número de subárboles n-arios asociados.
- Generalización de los árboles binarios. En este caso el orden de los subárboles no es importante.
- No es necesario saber cuál sub árbol es el primero o el último, sino simplemente saber que es un subárbol.



propiedades:

- Nodo: elemento del árbol
- Raíz: nodo inicial del árbol
- Hoja: nodo sin hijos
- Camino: nodos entre dos elementos incluyéndose
- Rama: camino entre la raíz y una hoja
- Altura: número de nodos en la rama más larga
- Peso: número de nodos en el árbol

Árbol Rojinegro

Que es?

Un árbol rojinegro es un árbol de búsqueda binario en el que cada nodo tiene un bit extra para almacenar su color.

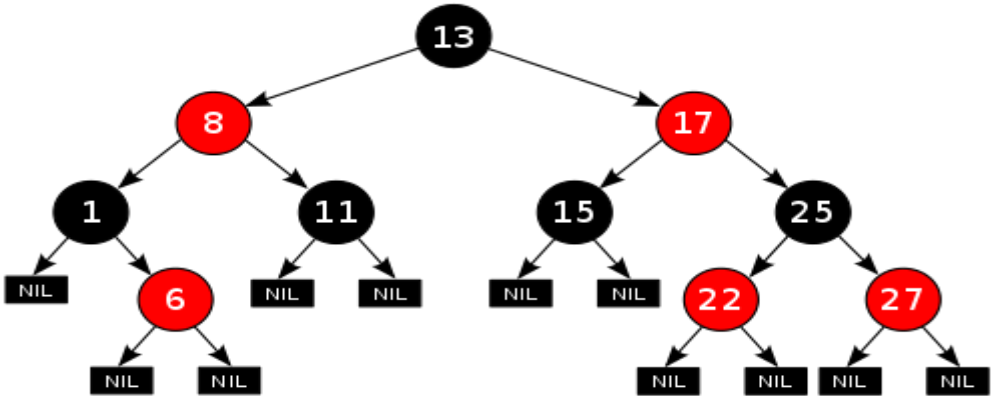
Cada nodo se registra con:

Color (rojo o negro).
Clave (la clave de búsqueda).
Dos punteros a los hijos (izquierdo y derecho).
Un puntero al padre. Si los punteros son nil, imaginamos que son punteros a nodos “externos” (hojas). Los demás nodos con claves se considerarán como nodos internos del árbol

Propiedades:

- Todo nodo es rojo o negro.
- La raíz es negra.
- Toda hoja (nil) es negra.
- Si un nodo es rojo, entonces sus hijos son negros.
- Cada camino de un nodo a sus hojas descendientes contienen el mismo número de nodos negros

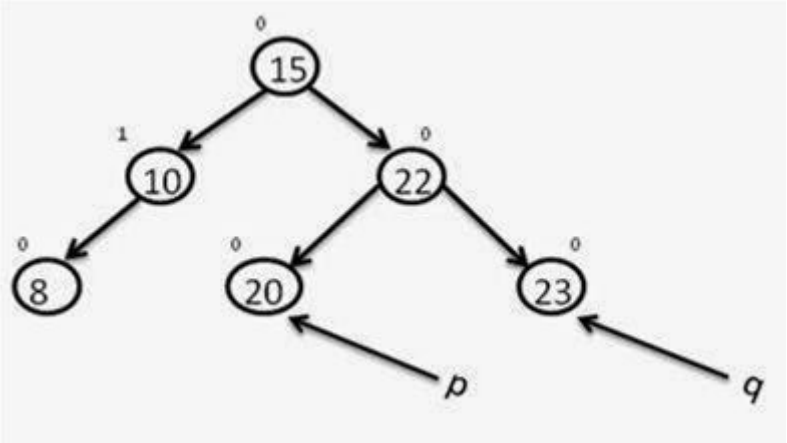
Este tipo de estructuras sirven para el almacenamiento de elementos entre los que exista una relación de orden. **La complejidad de la búsqueda en los árboles rojo negro es O (logn)**.



Árboles AVL:

Un árbol AVL es un tipo especial de árbol binario ideado por los matemáticos rusos AdelsonVelskii y Landis. Fue el primer árbol de búsqueda binario auto-balanceable que se ideó en el mundo.

Los árboles AVL están siempre equilibrados de manera que, para todos los nodos(Nodes), la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad O(log n). El factor de balanceo(utilizado en el curso de AED) puede ser almacenado directamente en cada nodo o ser computado a partir de las alturas de los subárboles.
Para conseguir esta propiedad de equilibrio, la inserción y el borrado de los nodos se ha de realizar de una forma especial. Si al realizar una operación de inserción o borrado se rompe la condición de equilibrio, hay que realizar una serie de rotaciones de los nodos.
Está balanceado en altura.
El tiempo de inserción y de búsqueda es de logn.



Fase 4: Transición de la formulación de ideas, a los diseños preliminares.

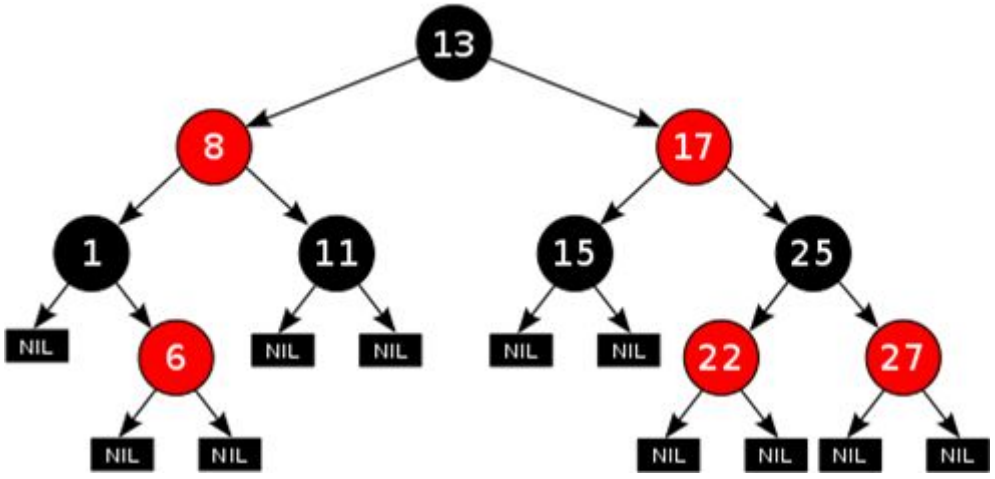
Se han escogido 3 estructuras de datos, debido que son las que más se adaptan a las necesidades del enunciado. La primera es Árboles Rojo y Negros, la segunda es Árboles AVL, y la tercera son Árboles de Búsqueda Binaria (ABB). Todas las estructuras se definen completamente a través de sus respectivos TADS, para que posteriormente se puedan implementar en el lenguaje de programación correspondiente.

1- ***TAD ARBOL ROJO Y NEGRO:***

El árbol rojinegro va a ser utilizado para ordenar los mercados de divisas por algunos de los criterios escogidos.

Nombre: TAD Árbol Rojo Y Negro.

Un árbol rojinegro se puede definir gráficamente como:



Invariantes:

- **La raíz siempre es negra.**
- **Un nodo puede ser de cualquier color (Rojo o Negro)**
- **Un hijo Rojo no puede tener un padre Rojo**
- **Todas las ramas del árbol tienen el mismo número de nodos negros.**

Operaciones:

Creadoras:

- **Crear Árbol Rojo y Negro: Árbol**

Descripción:

Construye un árbol vacío, se solicita memoria para almacenar elementos en el Árbol.

Postcondiciones: Árbol A ≠ Nill.

MODIFICADORAS:

- **Agregar (Element E): Árbol.**

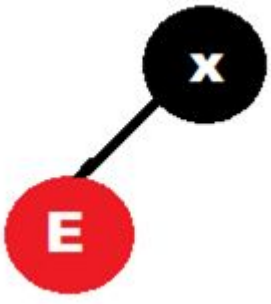
Precondiciones: Árbol A ≠ Nill; Elemento E a agregar ≠ Nill.

Descripción: Agrega a la Árbol A un nuevo elemento.

Postcondiciones:

Sea E el elemento a agregar en el Árbol A dado por:

Entonces:

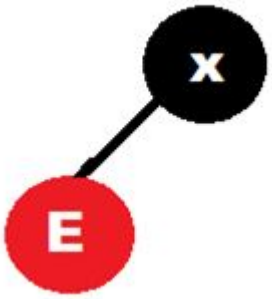


- **Eliminar (Element E): Árbol A.**

Precondiciones: Árbol A ≠ Nill, E (elemento a Eliminar) ≠ Nill.

Descripción: Elimina el elemento E, pasad por parámetro del árbol Rojo y Negro.

Postcondiciones:
Sea E el elemento a eliminar:



Entonces:

Rotar A la Izquierda o´ A la derecha (): void.

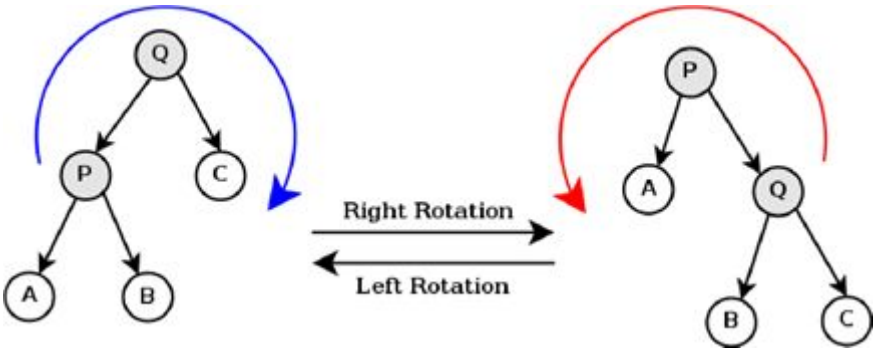
Precondiciones: Árbol A ≠ Nill

Descripción: Dado un puntero x, que representa la posición dentro del Árbol

A, la rotación cambia el hijo con el padre dejando a este último como “hijo” y conservando el orden dentro del árbol.

Postcondiciones:

Dado Árbol A entonces:



Analizadoras:

- **Buscar (Element E): Árbol A.**

Precondiciones: Árbol A ≠ Nill, E (elemento a Buscar) ≠ Nill.

Descripción: Busca un elemento E en el árbol A y lo retorna.

Postcondiciones:

Si el elemento E existe devuelve el Objeto que representa a E, En caso contrario retorna Nill.

2- *TAD ARBOL AVL:*

El árbol AVL va a ser utilizado para ordenar los Jugadores por algunos de los criterios escogidos.

Nombre: TAD ÁRBOL AVL

Invariantes:

- **El peso en cada punto del Árbol se calcula como, el número de nodos del camino derecho más largo menos el número de nodos del camino izquierdo más largo, y el resultado de esta resta debe estar entre -1 y 1.**

Operaciones:

Creadoras:

- **Crear Árbol AVL: Árbol**

Descripción:

Construye un árbol vacío, se solicita memoria para almacenar elementos en el Árbol.

Postcondiciones: Árbol A ≠ Nill.

Modificadoras:

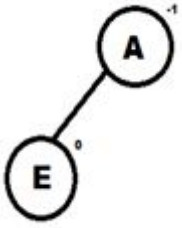
- **Agregar (Element E): Árbol.**

Precondiciones: Árbol A ≠ Nill; Elemento E a agregar ≠ Nill.
Descripción: Agrega a la Árbol A un nuevo elemento.

Postcondiciones:

Sea E el elemento a agregar en el Árbol A dado por:

Entonces:



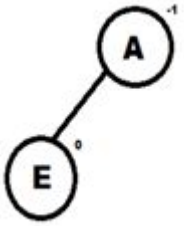
- **Eliminar (Element E): Árbol A.**

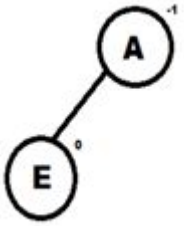
Precondiciones: Árbol A ≠ Nill, E (elemento a Eliminar) ≠ Nill.

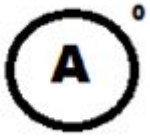
Descripción: Elimina el elemento E, pasa por parámetro del Árbol AVL.

Postcondiciones:

Sea E el elemento a eliminar:







Rotar A la Izquierda o´ A la derecha (): void.

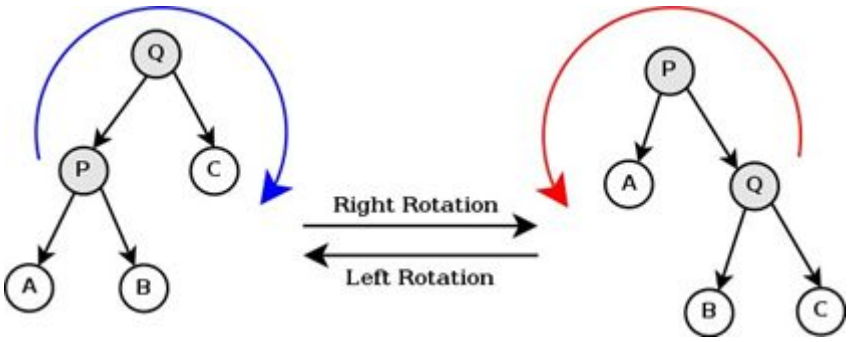
Precondiciones: Árbol A ≠ Nill

Descripción: Dado un puntero x, que representa la posición dentro del Árbol

A, la rotación cambia el hijo con el padre dejando a este último como “hijo” y conservando el orden dentro del árbol.

Postcondiciones:

Dado Árbol A entonces:



- **Analizadoras:**

Analizadoras:

- **Buscar (Element E): Árbol A.**

Precondiciones: Árbol A ≠ Nill, E (elemento a Buscar) ≠ Nill.

Descripción: Busca un elemento E en el árbol A y lo retorna.

Postcondiciones:

Si el elemento E existe devuelve el Objeto que representa a E, En caso contrario retorna Nill.

3- TAD Árbol binario de búsqueda:

El árbol Binario de Búsqueda va a ser utilizado para ordenar los Jugadores por algunos de los criterios escogidos.

Nombre: TAD ÁRBOL BINARIO DE BUSQUEDA

Invariantes:

-Dado un nodo x en el árbol, en una posición p, siempre los hijos derechos son mayores que su padre, y los hijos izquierdos menores que el.

Operaciones:

Creadoras:

- **Crear Árbol Binario De Búsqueda: Árbol**

Descripción:

Construye un árbol vacío, se solicita memoria para almacenar elementos en el Árbol.

Postcondiciones: Árbol A ≠ Nill.

Modificadoras:

- **Agregar (Element E): Árbol.**

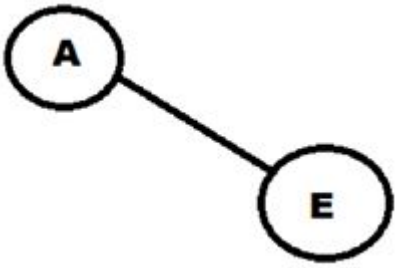
Precondiciones: Árbol A ≠ Nill; Elemento E a agregar ≠ Nill.

Descripción: Agrega a la Árbol A un nuevo elemento.

Postcondiciones:

Sea E el elemento a agregar al Árbol a en el siguiente estado.

Entonces si E es mayor que A:



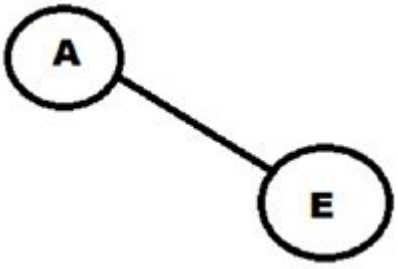
- **Eliminar (Element E): Árbol A.**

Precondiciones: Árbol A ≠ Nill, E (elemento a Eliminar) ≠ Nill.

Descripción: Elimina el elemento E, pasa por parámetro del Árbol Binario de Busqueda.

Postcondiciones:

Sea E el elemento a eliminar:



Entonces:

Analizadoras:

- **Buscar (Element E): Árbol A.**

Precondiciones: Árbol A ≠ Nill, E (elemento a Buscar) ≠ Nill.

Descripción: Busca un elemento E en el árbol A y lo retorna.
Postcondiciones:

Si el elemento E existe devuelve el Objeto que representa a E, En caso contrario retorna Nill.

Fase 5: Evaluación y selección de la mejor solución.

Se escogieron unos criterios para seleccionar qué estructuras se iban a implementar para modelar la solución del problema.

Criterios de selección de las estructuras de datos		si/no	
A	La estructura tiene complejidad lineal	3	1
B	La estructura utiliza una complejidad temporal menor que O(n) para sus búsquedas	5	1
C	La estructura es autobalanceable	5	1
D	La estructura permite utilizar n cantidad de datos	5	1
E	La estructura es estable	5	1

Fase 6: Preparación de informes y especificaciones.

Calificación de las estructuras escogidas						TOTAL
Criterios	A	B	C	D	E	
Estructura						
ABB	3	5	1	5	5	19
A-AVL	3	5	5	5	5	23
A-Roji-negro	3	5	5	5	5	23
Arbol N-Ario	1	1	1	5	5	13

Para concluir, después de realizar un amplio estudio y análisis de cada una de las estructuras presentadas anteriormente, y escogidas bajo unos criterios especificados en el punto anterior; seleccionamos las 2 siguientes estructuras: Árbol AVL y Árbol Rojinegro. Con estas se realizará la implementación de la solución, para así satisfacer los requerimientos del problema a tratar.